Southern New Hampshire University

CS 340 Final Project Guidelines and Rubric

Overview

The final project will encompass developing a web service using a software stack and implementing an industry-standard interface. Regardless of whether you choose to pursue application development goals as a pure developer or as a software engineer, creating and reusing software components will be a needed fundamental skill. Application programming interfaces (APIs) have become an industry-standard design pattern for enabling software component communication in a stacked development environment. In the final project, you will be taking advantage of and seeing firsthand the abilities of API calls to create a reusable software service. This service will enable other developers or software engineers to make use of your developed capabilities through an industry-standard web service API. In expanding your knowledge and use of APIs, you will in turn make use of the MongoDB API service to enable your application as a specialized information storage and retrieval service.

The project has **two milestones**, which will be submitted in **Modules Four and Five**. The final project will be submitted in **Module Seven**. Note, however, that the assignments in **Modules Two and Three** are crucial to building your MongoDB skills. Also note that throughout the course you will be practicing your newly learned skills on data sets that are different from your final project data set; when you complete your final project, you will demonstrate your mastery of your practiced final project skills using a new data set. For all of your assignments, make sure you are reviewing and incorporating instructor feedback before submitting your final project.

In this assignment, you will demonstrate your mastery of the following course outcomes:

- Communicate use of advanced programming technologies and associated programming operations effectively to stakeholders
- Use a higher level programming language for developing within a full stack environment
- Use advanced programming techniques that interface within a full stack environment
- Develop an application that integrates with current and emerging technologies in an effective and efficient manner

Prompt

As a newly hired software engineer for a financial services startup, you are tasked with creating a reporting service for basic stock market securities information. This reporting service will be used by other software engineers in the company for creating interactive web reports and dashboards.

Knowing that your service needs to be easily accessible, you have chosen to use the RESTful application programming interface (API) web-based protocol. Also knowing that your startup will more than likely focus in the future on various types of securities other than company stock, you decide to use MongoDB, a NoSQL document-storage system. MongoDB is supported by many languages, including Java and Python, via driver APIs. And since the RESTful API is already a part of many web application server frameworks using either Java or Python, you will only need to code and test the uniform resource identifier (URI) paths for the following functionality:

- Enable CRUD (create, read, update, and delete) operations specialized for stock market securities information.
- Select and present specific stock summary information info by a user-derived list of ticker symbols.
- Report a portfolio of five top stocks by a user-derived industry selection.
- Report a portfolio of possible stock investments for a selected company by similar industries.

To verify and test your RESTful web service, you will use a simple command line tool, such as curl, with a set of example URIs. Although the startup is a fast-moving organization, documentation is still a critical element for communication. Therefore, along with your RESTful API, you will provide user documentation in the form of explanations and screenshots.

Specifically, the following **critical elements** must be addressed:

I. **Collection Management:** In this section, you will create a database and create single or compound indexes. The database data set collections are preloaded in your final project tool.
   A. Utilize the mongoimport tool to **create a database** named "market" and a collection named "stocks," loaded with documents from the stocks.json file. Provide screenshots of the statements and the results of their execution.
   B. Assess the need for indexing as you formulate queries and, using the MongoDB shell, create any needed **single or compound indexes**. Provide screenshots of the statements and the results of their execution.
   C. For all of your screenshots, **explain** in detail each part of the associated MongoDB statements and their results to internal stakeholders. Be sure your explanations are logically organized and clearly communicated to meet the needs of the internal stakeholders.

II. **Document Manipulation:** In this section, you will add, update, and delete documents, making changes to the collection you created in the previous section. Provide source code in a text file for the functions you will create below.
   A. **Insert** new key-value pairs into documents using appropriate MongoDB statements. Specifically, create a function or method in Python or Java that will read from a file or standard input stream a value pair stream in JSON notation and insert this document into the stocks collection. You will also need to create a simple application scaffold for testing your function or method. Provide screenshots of the results of their execution.
   B. **Update** existing documents using appropriate MongoDB statements. Specifically, create a function or method in Python or Java that will update the document "Volume" key-value pair identified by the string input stock ticker symbol "Ticker" and numerical input "Volume" value of your choice greater than zero. The function or method will update the document "Volume" key-value pair identified by the given ticker symbol and a new "Volume" value of your choice greater than zero. You will also need to create a simple main application to call your function. Provide screenshots of the results of their execution.
   C. **Delete** existing documents using appropriate MongoDB statements. Specifically, create a function or method in Python or Java that will take as input a stock ticker symbol "Ticker." The function or method will remove the document identified by the given ticker symbol. For example, use the ticker symbol "BRLI." You will also need to create a simple application scaffold for testing your function or method. Provide screenshots of the results of their execution.

D. For all your screenshots, provide **explanations** of each part of the associated MongoDB statements and their results. Be sure your explanations are logically organized and clearly communicated to meet the needs of the internal stakeholders.

III. **Document Retrieval:** In this section, you will create code to query the collection to retrieve information about the application. Provide source code in a text file for the functions you will create below.

A. Retrieve documents from collections by using the appropriate **find statement arguments**.

i. Specifically, create a function or method in Python or Java that will take as inputs **numerical values** for low and high. The function or method will find documents for which the "50-Day Simple Moving Average" is between the low and high values and return the count of the number of documents found. You will also need to create a simple main application to call your function. Provide screenshots of the results of their execution.

ii. Additionally, create a function or method in Python or Java that will take as input a **string**. The function or method will find documents for which the input string matches the document key "Industry" and returns the list of ticker symbols found to match that industry. For example, use the industry string "Medical Laboratories & Research." Again, you will also need to create a simple main application to call your function. Provide screenshots of the results of their execution.

B. Write MongoDB **aggregation pipeline statements** that transform documents into aggregated results using multiple pipeline stages as appropriate. Specifically create a function or method in Python or Java that will take as input a string. The function or method will find documents for which the input string matches the document key "Sector" and returns the total outstanding shares grouped by document key "Industry." Examples of sector string inputs are "Healthcare," "Basic Materials," and so on. You will also need to create a simple main application to call your function. Provide screenshots of the results of their execution.

C. For all of your screenshots, provide **explanations** of each part of the associated MongoDB statements and their results. Be sure your explanations are logically organized and clearly communicated to meet the needs of the internal stakeholders.

IV. **Advanced Programming Project:** In this section, you will develop a web service application to implement a RESTful application programming interface (API) for a MongoDB database. Provide source code in a text file (either **.java** or **.py**) for your complete web service application that encompasses all of the functionality below.

A. Develop a **RESTful API** using a Python or Java web services framework for a MongoDB collection of stock market summary data, ensuring your code is functional, reusable, concise, and commented.

B. Enable specific **CRUD functionality** in a developed RESTful API framework. Use the example URIs linked in the prompt to test and validate your framework. Provide screenshots of the code and its execution, ensuring your code is functional, reusable, concise, and commented.

C. In your RESTful API, enable the following functionality (advanced querying), ensuring your code is functional, reusable, concise, and commented:

i. Select and present specific **stock summary information** by a user-derived list of ticker symbols. Provide screenshots of the code and its execution.

ii. Report a portfolio of **five top stocks** by a user-derived industry selection. Provide screenshots of the code and its execution.

Southern New Hampshire University

Milestones

Milestone One: *Implementing CRUD Operations in Python or Java*
In **Module Four**, you will implement the fundamental operations of create, read, update, and delete (CRUD) in either Python or Java. You will use the language-specific MongoDB driver to create CRUD functional access to your document collection. This milestone will be graded with the **Milestone One Rubric**.

Milestone Two: *Implementing a Basic RESTful Web Service*
In **Module Five**, you will implement a basic RESTful web service using either Python or Java. You will use the language-specific web services framework to implement the RESTful service. This milestone will be graded with the **Milestone Two Rubric**.

Final Submission: *RESTful API and User Documentation*
In **Module Seven**, you will submit your final project. It should be a complete, polished artifact containing **all** of the critical elements of the final project. It should reflect the incorporation of feedback gained throughout the course. This submission will be graded with the **Final Project Rubric**.

Final Project Rubric

Guidelines for Submission: Your RESTful API and user documentation will be submitted in two parts. The user documentation (screenshots and explanations) must be 3 to 5 pages in length and must be written in APA format. Use double spacing, 12-point Times New Roman font, and one-inch margins. You must also submit your code in either **.java** or **.py** format.

Critical Elements	Exemplary	Proficient	Needs Improvement	Not Evident	Value
Collection Management: Create a Database		Creates database using appropriate MongoDB statements and provides screenshots of the statements and results of execution (100%)	Creates database using appropriate MongoDB statements but does not provide screenshots, or not all of the statements are appropriate (55%)	Does not create collections (0%)	4.75
Collection Management: Single or Compound Indexes		Creates single or compound indexes for the collections using appropriate MongoDB statements and provides screenshots of the statements and results of execution (100%)	Creates single or compound indexes for the collections but does not provide screenshots, or not all of the statements are appropriate (55%)	Does not create single or compound indexes (0%)	4.75

Collection Management: Explanation	Meets "Proficient" criteria and demonstrates discerning insight into the needs of the stakeholders and how best to communicate with them (100%)	Explains each part of the associated MongoDB statements and their results to internal stakeholders, ensuring explanations are logically organized and clearly communicated to meet the needs of stakeholders (85%)	Explains each part of the associated MongoDB statements and their results to internal stakeholders, but not all are logically organized or clearly communicated to meet the needs of stakeholders (55%)	Does not explain each part of the associated MongoDB statements and their results (0%)	7.9
Document Manipulation: Insert		Inserts new key-value pairs into documents using appropriate MongoDB statements and provides screenshots of the statements and results of execution (100%)	Inserts new key-value pairs into documents but does not provide screenshots, or not all of the statements are appropriate (55%)	Does not insert new key-value pairs (0%)	4.75
Document Manipulation: Update		Updates existing documents using appropriate MongoDB statements and provides screenshots of the statements and results of execution (100%)	Updates existing documents but does not provide screenshots, or not all of the statements are appropriate (55%)	Does not update existing documents (0%)	4.75
Document Manipulation: Delete		Deletes existing documents using appropriate MongoDB statements and provides screenshots of the statements and results of execution (100%)	Deletes existing documents but does not provide screenshots, or not all of the statements are appropriate (55%)	Does not delete existing documents (0%)	4.75
Document Manipulation: Explanations	Meets "Proficient" criteria and demonstrates discerning insight into the needs of the stakeholders and how best to communicate with them (100%)	Explains each part of the associated MongoDB statements and their results to internal stakeholders, ensuring explanations are logically organized and clearly communicated to meet the needs of stakeholders (85%)	Explains each part of the associated MongoDB statements and their results to internal stakeholders, but not all are logically organized or clearly communicated to meet the needs of stakeholders (55%)	Does not explain each part of the associated MongoDB statements and their results (0%)	7.9

Document Retrieval: Find Statement Arguments: Numerical Values		Retrieves documents from collections using the appropriate find statement arguments by creating a function or method that will take as inputs numerical values for low and high and provides screenshots of the statements and results of execution (100%)	Retrieves documents from collections using the appropriate find statement arguments by creating a function or method that will take as inputs numerical values for low and high, but does not provide screenshots, or not all statements are appropriate (55%)	Does not retrieve documents (0%)	7.9
Document Retrieval: Find Statement Arguments: String		Retrieves documents from collections using the appropriate find statement arguments by creating a function or method that will take as input a string, and provides screenshots of the statements and results of execution (100%)	Retrieves documents from collections using the appropriate find statement arguments by creating a function or method that will take as input a string, but does not provide screenshots, or not all statements are appropriate (55%)	Does not retrieve documents (0%)	7.9
Document Retrieval: Aggregation Pipeline Statements		Writes MongoDB aggregation pipeline statements that transform documents into aggregated results using multiple pipeline stages as appropriate and provides screenshots of the statements and results of execution (100%)	Writes MongoDB aggregation pipeline statements that transform documents into aggregated results using multiple pipeline stages as appropriate but does not provide screenshots, or not all statements are appropriate (55%)	Does not write MongoDB aggregation pipeline statements (0%)	7.9
Document Retrieval: Explanations	Meets “Proficient” criteria and demonstrates discerning insight into the needs of the stakeholders and how best to communicate with them (100%)	Explains each part of the associated MongoDB statements and their results to internal stakeholders, ensuring explanations are logically organized and clearly communicated to meet the needs of stakeholders (85%)	Explains each part of the associated MongoDB statements and their results to internal stakeholders, but not all are logically organized or clearly communicated to meet the needs of stakeholders (55%)	Does not explain each part of the associated MongoDB statements and their results (0%)	7.9

Advanced Programming Project: RESTful API	Meets "Proficient" criteria and develops a RESTful API demonstrating a sophisticated use of Python or Java frameworks and functional, reusable, concise, and commented coding practices (100%)	Develops a RESTful API using a Python or Java web services framework for a MongoDB collection, ensuring code is functional, reusable, concise, and commented (85%)	Develops a RESTful API using a Python or Java web services framework, but RESTful API is not functional, reusable, concise, or commented (55%)	Does not develop a RESTful API (0%)	5.9
Advanced Programming Project: CRUD Functionality	Meets "Proficient" criteria and enables CRUD functionality demonstrating a sophisticated use of functional, reusable, concise, and commented coding practices (100%)	Enables specific CRUD functionality in developed RESTful API framework, ensuring code is functional, reusable, concise, and commented, uses URIs to test and validate framework, and provides screenshots of the code and its execution (85%)	Enables specific CRUD functionality in developed RESTful API framework but does not use URIs to test and validate framework, does not provide screenshots, or code is not functional, reusable, concise, or commented (55%)	Does not enable specific CRUD functionality (0%)	5.9
Advanced Programming Project: Stock Summary Information	Meets "Proficient" criteria and demonstrates a sophisticated practice of advanced querying (100%)	Selects and presents specific stock summary information by a user-derived list of ticker symbols and provides screenshots of the code and its execution (85%)	Selects and presents specific stock summary information but does not provide screenshot, or query is not appropriate (55%)	Does not select or present specific stock summary information (0%)	5.9
Advanced Programming Project: Five Top Stocks	Meets "Proficient" criteria and demonstrates a sophisticated practice of advanced querying (100%)	Reports a portfolio of five top stocks by a user-derived industry selection and provides screenshots of the code and its execution (85%)	Reports a portfolio of five top stocks but does not provide screenshot, or query is not appropriate (55%)	Does not report a portfolio of five top stocks (0%)	5.9
Articulation of Response	Submission is free of errors related to citations, grammar, spelling, syntax, and organization and is presented in a professional and easy-to-read format (100%)	Submission has no major errors related to citations, grammar, spelling, syntax, or organization (85%)	Submission has major errors related to citations, grammar, spelling, syntax, or organization that negatively impact readability and articulation of main ideas (55%)	Submission has critical errors related to citations, grammar, spelling, syntax, or organization that prevent understanding of ideas (0%)	5.25
				Total	**100%**